Application of Rank Correlation, Clustering and Classification in Information Security

Gleb Beliakov[1], John Yearwood[2], Andrei Kelarev[1]
[1]School of Information Technology, Deakin University, 221 Burwood Hwy, Burwood 3125, Australia
[2]Graduate School of SITE, University of Ballarat, P.O. Box 663, Ballarat, Victoria 3353, Australia
Email: gleb@deakin.edu.au, andrei.kelarev@yahoo.com.au, j.yearwood@ballarat.edu.au

***Abstract*— This article is devoted to experimental investigation of a novel application of a clustering technique introduced by the authors recently in order to use robust and stable consensus functions in information security, where it is often necessary to process large data sets and monitor outcomes in real time, as it is required, for example, for intrusion detection. Here we concentrate on a particular case of application to profiling of phishing websites. First, we apply several independent clustering algorithms to a randomized sample of data to obtain independent initial clusterings. Silhouette index is used to determine the number of clusters. Second, rank correlation is used to select a subset of features for dimensionality reduction. We investigate the effectiveness of the Pearson Linear Correlation Coefficient, the Spearman Rank Correlation Coefficient and the Goodman–Kruskal Correlation Coefficient in this application. Third, we use a consensus function to combine independent initial clusterings into one consensus clustering. Fourth, we train fast supervised classification algorithms on the resulting consensus clustering in order to enable them to process the whole large data set as well as new data. The precision and recall of classifiers at the final stage of this scheme are critical for effectiveness of the whole procedure. We investigated various combinations of several correlation coefficients, consensus functions, and a variety of supervised classification algorithms.**

***Index Terms*— consensus functions; clustering; classification; phishing websites**

I. Introduction

This article deals with the experimental investigation of various combinations of rank correlation coefficients, consensus functions and supervised classification algorithms for the profiling of phishing websites. There are many clustering algorithms known in the literature. However, their outcomes depend on the random selection of initial seeds. Our approach has been designed to enable the application of consensus functions, since they can be used to increase stability and robustness of the obtained clusterings. The readers are referred to Section V for preliminaries and background information on consensus functions, see also [1] and [2] for additional references and examples of recent results.

This paper is based on "An Application of Novel Clustering Technique for Information Security" by G. Beliakov, J. Yearwood, and A. Kelarev, which appeared in the Proceedings of the Applications and Techniques in Information Security (ATIS2011), Melbourne, Australia, November 2011.

The data sets in information security are very large and often require real-time monitoring, which is necessary, for example, for intrusion detection. This is why direct applications of sophisticated consensus functions in this area are computationally expensive. To overcome this difficulty, in [1] the authors have introduced a new approach combining consensus functions with fast supervised classification algorithms.

The present paper is devoted to experimental investigation of the effectiveness of this technique for the new application to profiling of phishing websites. This application has not been considered before and belongs to the information security domain that has been actively investigated recently, as described by the Anti-Phishing Working Group [3] and OECD Task Force on Spam [4], see also [1] and [2]. We hope that the outcomes of our experiments can also provide guidance for choosing directions of future studies in other branches of information security.

This novel technique makes it possible to utilize slow and most reliable consensus functions at the initial stages to obtain more robust clusterings. On the other hand, it increases the speed of processing the whole large data set and new samples by incorporating fast classification algorithms in the final stage.

The paper is organised as follows. An outline of the combined approach to clustering is given in Section II. Section III is devoted to the preprocessing of data and extraction of features for clustering algorithms. Section IV contains a brief outline of clustering algorithms applied to obtain an initial clustering ensemble for a small randomized sample of data. Section V contains background information on consensus functions and concise preliminaries on graph formulations and heuristics used to combine the ensemble into one final consensus clustering. Section VI deals with the supervised classification algorithms trained on the consensus clustering. Experimental results are summarized in Section VII. Section VIII presents the conclusions.

II. Outline of the combined approach to clustering

We investigated various combinations of rank correlation coefficients and consensus functions in conjunction with fast supervised classification algorithms. This

approach to clustering has several steps or stages. First, a variety of independent clustering algorithms are applied to a randomized sample of data. Second, rank correlation is used to select a subset of features for dimensionality reduction. Our experiments investigated the effectiveness of several correlation coefficients in this procedure. Third, a consensus function is used to combine these independent clusterings into one common consensus clustering. In fact, we investigated the effectiveness of several consensus functions in this scheme. Fourth, the consensus clustering of the randomized sample is used as a training set to train fast supervised classification algorithms. Finally, these fast classification algorithms can be applied to classify the whole large data set.

Our experiments investigated this approach for a particular case of profiling of phishing websites. Algorithms for classification and clustering of phishing emails and websites have been actively investigated recently and many valuable results have been obtained, as discussed for example, by the Anti-Phishing Working Group [3] and OECD Task Force on Spam [4], see also [1] and [2] for examples of recent results. Phishing usually involves acts of social engineering attempting to extract confidential details by sending emails with false explanations urging users to provide private information that will be used for identity theft. The users are then directed to a phishing website, where they are asked to enter personal details, such as credit card numbers, tax file numbers, bank account numbers and passwords. For more comprehensive information on phishing the readers are referred to the Anti-Phishing Working Group [3] and OECD Task Force on Spam [4].

Our experiments compared the effectiveness of several correlation coefficients, CBGF, HBGF and IBGF consensus functions and investigated the performance of their various combinations with fast classification algorithms incorporated in this scheme.

III. Feature Extraction

A large data set of phishing websites has been supplied by the industry partners of the Centre for Informatics and Applied Optimization at the University of Ballarat. Analogous data sets are available from the downloadable databases at the PhishTank [5].

A flexible preprocessing and feature extraction system has been implemented in Python for the purposes of this investigation. It has been used to extract features describing the content and html structure of the websites.

A collection of features was extracted by considering each website according to the bag-of-words model, where it is viewed as an unordered collection of words and the grammar, structure and word order are ignorred. We used the *term frequency–inverse document frequency* word weights, or TF-IDF weights, to select features for the clusterings. These weights are well known in feature extraction for text categorization [6], [7], [8]. They are defined using the following concepts and notation. Suppose that we are extracting features from a data set E, which consists of $|E|$ websites. For a word w and a website m, let $\mathrm{N}(w,m)$ be the number of times w occurs in m. Suppose that a collection $T = \{t_1, \ldots, t_k\}$ of terms $t_1, \ldots, t_k$ is being looked at. The *term frequency* of a word $w \in T$ in a website m is denoted by $\mathrm{TF}(w,m)$ and is defined as the number of times w occurs in m, normalized over the number of occurrences of all terms in m:

$$\mathrm{TF}(w,m) = \frac{\mathrm{N}(w,m)}{\sum_{i=1}^{k} N(t_i,m)}. \tag{1}$$

The *document frequency* of the word w is denoted by $\mathrm{DF}(w)$ and is defined as the number of websites in the given data set where the word w occurs at least once. The *inverse document frequency* is used to measure the significance of each term. It is denoted by $\mathrm{IDF}(w)$ and is defined by the following formula

$$\mathrm{IDF}(w) = \log\left(\frac{|E|}{\mathrm{DF}(w)}\right). \tag{2}$$

The *term frequency–inverse document frequency* of a word w in website m, or TF-IDF weight of w in m is defined by

$$\text{TF-IDF}(w,m) = \mathrm{TF}(w,m) \times \mathrm{IDF}(w). \tag{3}$$

We collected a set of words with highest TF-IDF scores in all websites of the data set. For each website, the TF-IDF scores of these words in the website were determined. These weights and additional features were assembled in a vector. In order to determine the TF-IDF scores we used Gensim, a Python package for vector space modelling of text documents using NumPy and SciPy. In addition, we used features reflecting the html structure of the websites and links to different URL domains or numeric IP addresses.

We have also incorporated several features related to the structure of the websites, including

- the number of images,
- sizes and quality of images,
- hidden fields or graphics,
- full HTML substitution in the links,
- links to webpages in other domains,
- inline embedding of scripts,
- loading external scripting code,
- unicode-obfuscated URLs,
- URLs beginning with IP addresses in links.

These features were assembled in an algebraic vector space model representing the data set. A number of independent initial clusterings were then obtained for the feature vectors of the websites in the sample using the following clustering algorithms.

IV. Independent Initial Clusterings

Looking at the features extracted as described in Section III, we used four clustering algorithms implemented in WEKA, SimpleKMeans, Cobweb, EM and FarthestFirst, and obtained an ensemble of independent initial clusterings $C = \{C^{(1)}, C^{(2)}, \ldots, C^{(k)}\}$, where, for each

clustering $C^{(i)}$, the whole data set D is a disjoint union of the classes in this clustering so that

$$C^{(i)} = \{C_1^{(i)}, C_2^{(i)}, \ldots, C_{k_i}^{(i)}\} \text{ and} \quad (4)$$

$$D = C_1^{(i)} \dot\cup C_2^{(i)} \dot\cup \ldots \dot\cup C_{k_i}^{(i)}, \quad (5)$$

for all $i = 1, \ldots, k$.

SimpleKMeans is the classical k-mean clustering algorithm described, for example, in [9], Section 3.3.2, and [10], Section 4.8. This algorithm randomly chooses k websites as centroids of clusters at the initialization stage. Every other website is allocated to the cluster of its nearest centroid. After that each iteration finds new centroids of all current clusters as a mean of all members of the cluster. This is equivalent to finding the point such that the sum of all distances from the new centroid to all other sequences in the cluster is minimal. Then the algorithm reallocates all points to the clusters of the new centroids. It proceeds iteratively until the centroids stabilize. We used SimpleKMeans with the default Euclidean distance.

The outcomes of the k-means algorithm often depend on the initial selection of the very first centroids. The outcome of the SimpleKMeans in the WEKA implementation depends on the value of the input parameter "seed". To overcome the dependence of the outcome on the random choice of this parameter we run it with 10 random selections of the "seed", as recommended in [11].

Cobweb is the WEKA implementation of the Cobweb and Classit clustering algorithms described in [12] and [13], respectively. EM is the expectation maximisation algorithm in WEKA. It determines the probability of each instance belonging to each of the clusters. It can be used to assign every instance to the cluster where it belongs with the highest probability. FarthestFirst is a WEKA implementation of the clustering algorithm described in [14]. Cobweb, EM, FarthestFirst and SimpleKMeans produce clusterings given a fixed number of clusters as an input parameter.

In order to determine the appropriate number of clusters we used Silhouette index. The *Silhouette index* of a clustering is a robust measure of the quality of the clustering introduced in [15]. The Silhouette index $\mathrm{SI}(x)$ of each observation x is defined as follows. If x is the only point in its cluster, then $\mathrm{SI}(x) = 0$. Denote by $a(x)$ the average distance between x and all other points of its cluster. For any other cluster C, let $d(x, C)$ be the average distance between x and all points of C. The minimum

$$b(x) = \min\{d(x, C) \,:\, x \notin C\} \quad (6)$$

is the distance from x to its nearest cluster C to which x does not belong. Finally, put

$$\mathrm{SI}(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \quad (7)$$

The Silhouette index of the whole clustering is found as the average index over all observations. The Silhouette index always belongs to $[-1, 1]$. The partition with highest Silhouette index is regarded as optimal.

For each initial clustering algorithm and each value of the "seed", we repeated it several times increasing the number of clusters, as recommended in [15]. The clustering with the best Silhouette index was included in the set of initial clusterings to be processed by consensus clustering algorithm at the next stage. The same procedure of determining the number of clusters was applied for other initial clustering algorithms too.

All these initial clustering algorithms can process our data without any additional data transformations or encoding. The outcomes of all of these clustering algorithms often depend on the initial random selections made during the start of their iterations. A standard method is to run them for several random selections of initial parameters, as in [11]. In WEKA, the outcomes of these algorithms depend on their input parameter "seed". We run each of these algorithms for 10 random selections of the "seed" and obtained a total of 40 initial clusterings. This provided sufficient input for the consensus clustering algorithms considered in the next section. Thus, we have used multiple start versions of the Cobweb, EM, FarthestFirst and SimpleKMeans, which could process our sample directly and produced sufficient input for the next stage of our approach.

V. Consensus Functions for Ensemble Clusterings

The following three consensus functions have been applied:

CBGF – Cluster-Based Graph Formulation,
HBGF – Hybrid Bipartite Graph Formulation,
IBGF – Instance-Based Graph Formulation

The final consensus clustering obtained by these consensus functions was used to train fast classification algorithms. It is therefore natural to assume that a consensus function performs better in our scheme, if the supervised classification algorithms are able to produce higher precision and recall at the final stage.

Cluster-Based Graph Formulation, CBGF, is a graph-based consensus function. It defines a complete weighted undirected graph on the set of vertices consisting of all the given clusters. The weight of each edge of this graph is determined by a measure of similarity of the clusters corresponding to the vertices. Namely, for two clusters C' and C'' the weight of the edge (C', C'') can be set equal to

$$w((C', C'')) = \frac{|C' \cap C''|}{|C' \cup C''|}, \quad (8)$$

known as the *Jaccard index* or *Jaccard similarity coefficient*, see [16], Chapter 2. In order to ensure that clusters that have a lot of elements in common are grouped together, the edges with lowest weights are then eliminated by applying a graph partitioning algorithm. Each element is then allocated to the new final cluster where it occurs most frequently.

Hybrid Bipartite Graph Formulation, HBGF, is a consensus function proposed in [17] and based on a bipartite

graph. It has two sets of vertices: clusters and elements of the data set. A cluster C and an element d are connected by an edge in this bipartite graph if and only if d belongs to C. An appropriate graph partitioning algorithm is then applied to the whole bipartite graph. The final clustering is determined by the way it partitions all elements of the data set. We refer to [17]–[19] for more details.

Instance-Based Graph Formulation, IBGF, is also a consensus function based on a complete undirected weighted graph. Vertices of the graph are all elements of the data set. The edge (d', d'') has weight given by the formula

$$w((d', d'')) \quad = \sum_{i=1,\ldots,k;\ C_i(d')=C_i(d'')} 1/k,$$

where $C_i(x)$ stands for the cluster containing x in the i-th clustering. This means that $w((d', d''))$ is the proportion of clusterings where the clusters of d' and d'' coincide. Then IBGF applies an appropriate graph partitioning algorithm to divide the graph into classes. These classes determine clusters of the final consensus clustering.

We used METIS graph partitioning software described in [20]. The weights of edges in the input files of METIS must all be strictly greater than zero, which means that it can handle only complete weighted graphs. In order to apply it to a bipartite graphs, we had to set the weights of all edges not present in the graph to 1 and to rescale the weight of all other edges by multiplying them with a constant to make them larger than 10,000. This ensured that METIS removed all nonexistent edges from the graph and then continued analysing the resulting bipartite graph.

We used rank correlation to select the most essential features for use in consensus functions. It ranks all features with respect to their relevance and importance to the problem. We investigated four well-known measures: the Pearson Linear Correlation Coefficient, the Spearman Rank Correlation Coefficient, Kendall Rank Correlation Coefficient, KRCC, and the Goodman–Kruskal Correlation Coefficient. They can help remove irrelevant features with almost zero correlation to the cluster labels. As a result the redundancy among similar selected features is reduced. Other methods, such as symmetrical uncertainty [21] and asymmetric dependency coefficient [22] are also usable for feature ranking to measure the relevance of the features.

First, we used the Pearson Linear Correlation Coefficient, PLCC, also known as Pearson's Product-Moment Correlation Coefficient, [8]. It is well known that it is helpful in various situations and has low complexity [21], [23]. The PLCC is calculated to assess the correlation between the features and their class labels. It is defined by the following formula [23]:

$$\rho(f_r, I) = \frac{\mathrm{cov}(f_r, I)}{\sigma(f_r)\sigma(I)}, \tag{9}$$

where $\sigma(I)$ is the standard deviation of the labels of instances and the covariance $\mathrm{cov}(f_r, I)$ between f_r and I is defined by

$$\mathrm{cov}(f_r, I) = \frac{\sum_{i=1}^{n}(f'_r - d_{ir})(I' - I_i)}{n} \tag{10}$$

where I_i is the label of the instance d_i and I' is the mean of labels of instances. The standard deviation $\sigma(f_r)$ can be calculated as

$$\sigma(f_r) = \sqrt{\frac{\sum_{i=1}^{n}(f'_r - d_{ir})^2}{n}} \tag{11}$$

and f'_r is the mean of the feature f_r,

$$f'_r = \frac{\sum_{i=1}^{n}(d_{ir})}{n}. \tag{12}$$

Second, we used the Spearman Rank Correlation Coefficient, SRCC, also known as Spearman's Rho [24], [25], [8]. It assesses how well the relationship can be described using a monotonic function, which does not have to be linear. The SRCC ϱ is a measure of association based on the ranks of the data values. It is given by the formula

$$\varrho = \frac{\sum(R_i - \overline{R})(S_i - \overline{S})}{\sqrt{\sum(R_i - \overline{R})\sum(S_i - \overline{S})}}, \tag{13}$$

where R_i is the rank of the i-th x-value, S_i is the rank of the i-th y-value, $\overline{R}$ is the mean of the ranks of x-values, and $\overline{S}$ is the mean of the ranks of y-values. The values of ϱ belong to the segment $[-1; 1]$. Values close to 1 indicate that there is a good correlation (described by a monotonically increasing function). First, we obtained initial clusterings for the small randomized sample and all original features as described in Section IV. Then we used these initial clusterings to find the Spearman Rank Correlation Coefficients. For each numerical feature, we numbered all clusters according to the mean value of this feature for all instances of the cluster, and after that ranked all values of the feature and the cluster numbers. Numbering clusters in the order of the mean values of the feature for all instances of each cluster is essential, since it ensures that we only have to look at values of SRCC close to 1 in our case. Having found the SRCC for each feature, we ordered the original features by the values of their Spearman Rank Correlation Coefficients. The features with higher values were selected for the next stage of our procedure.

Third, we used Kendall Rank Correlation Coefficient, KRCC, also known as Kendall's Tau, [24], [25], [8]. Our experiments have shown that it produces outcomes very similar to the SRCC, and so in this paper we include only the tables of the precision and recall obtained with the SRCC.

Fourth, we used the Goodman–Kruskal Correlation Coefficient, GKCC, also known as the Goodman–Kruskal's Gamma, [24], [25], [8]. It is defined as the difference between the number of concordant pairs C and the

number of discordant pairs D of the two rankings, as a proportion of all pairs, ignoring ties:

$$G = (C - D)/(C + D). \quad (14)$$

GKCC tests for a weak monotonicity between the two rankings. The value of GKCC ranges between +1 to -1, and it is equal to 0 for independent variables.

We ranked all the preliminary variables according to the values of their rank correlation coefficients. Different testing data sets or clustering algorithms will produce different ranking lists of the preliminary variables. The principle is that, the higher the ranking of the feature, the more relevant it is to the clustering result. This means that not all of the features make the same contribution to the clustering result. The least important features can be regarded as redundant features and can be removed. The quality of the clusters can be improved by eliminating the influence of the redundant features, and the efficiency of clustering algorithm can be increased by reducing dimensionality and removing irrelevant features.

In order to determine the appropriate number of clusters for the final consensus clustering we used Silhouette index described in Section IV. We ran each consensus clustering increasing the number of clusters from 2 to 30. The final consensus clustering with the best Silhouette index was then regarded as the final output of the whole process, as illustrated in Figure 1.

VI. Supervised Classification Algorithms

We have compared the performance of these three consensus functions and their combinations with several supervised classification algorithms. The resulting consensus clustering described in Section V was used to train supervised classification algorithms. We investigated the performance of all classifiers implemented in WEKA, and have included in the tables of this paper the outcomes of the following algorithms, which worked well in our scheme:

- BayesNet – Bayes Network learning algorithm K2, [26].
- DecisionTable, a decision table majority classifier [27].
- IBk, a k-nearest neighbours classifier selecting an appropriate value of k based on cross-validation, [28].
- J48 classifier generating a C4.5 decision tree, [29].
- JRip classifier implementing a propositional rule learner RIPPER, [30].
- HyperPipes, a simple and fast classifier, [26].
- LibLINEAR, a library for large linear classification, [31].
- LibSVM, a library for Support Vector Machines, [32]. It implements an SMO-type algorithm proposed by [33].
- NaiveBayes classical algorithm, [10], [34].
- PART classifier generating decision list based on partial C4.5 decision trees and separate-and-conquer, [35].
- RBFNetwork implementing a normalized Gaussian radial basis function network, [26].
- Ridor – a ripple down rule classifier, [26].
- SMO classifier using Sequential Minimal Optimization for training a support vector classifier, [36]–[38],
- VFI – voting feature intervals classification due to [39].

More information on these algorithms is given by [10], [26], [28], [29], [33]–[35], [37], [39], [40].

The performance of the SMO, LibSVM and LibLINEAR depends on the SVM type, the kernel and several numerical parameters. We have considered all types of SVMs and kernels in SMO, LibSVM and LibLINEAR that could handle the format of our data without additional preprocessing. For each of these cases, we used the optimization procedure explained in [41]. More advanced optimization techniques presented in [42] can also be applied here.

VII. Experimental results

We have undertaken experimental investigation of the novel approach to clustering for a randomized sample of phishing websites. Our experiments have compared all combinations of the PLCC, SRCC and KRCC correlation coefficients with CBGF, HBGF and IBGF consensus functions and classification algorithms listed above for a randomized sample of 1024 websites. We used tenfold cross validation to evaluate the weighted average precision and recall of these classification algorithms comparing them with the classes of the corresponding consensus clustering.

The results of our experiments are summarized in Tables V, VI, VII and VIII. The precision and recall for all choices of kernels of the SMO, LibSVM and LibLINEAR classifiers are assembled in Tables V and VI. Their best results have been also included in Tables VII and VIII for convenience of the readers. The outcomes show that the combination of the Goodman–Kruskal Correlation Coefficient, the Hybrid Bipartite Graph Formulation consensus function, and the Sequential Minimal Optimization classifier with the polynomial kernel achieved the best precision and recall in these experiments.

VIII. Conclusion

This article investigated a novel approach to clustering of information security data sets and presented experimental results for the particular case of application to profiling phishing websites. Our method is based on combining rank correlation coefficients and reliable consensus functions with fast supervised classification algorithms. First, we applied a variety of independent clustering algorithms to a randomized sample of data. Silhouette index was used to determine the number of clusters for these algorithms. Second, rank correlation was used to select a subset of features for dimensionality reduction. Our experiments compared the effectiveness of four correlation coefficients in this procedure: Pearson Linear Correlation Coefficient, PLCC, Spearman Rank Correlation Coefficient,

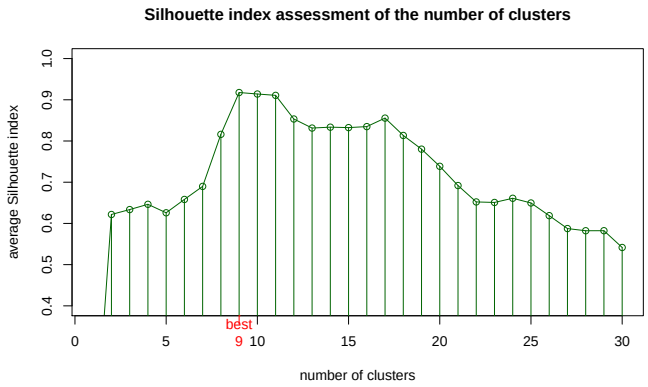

Figure 1. Silhouette indices of the final consensus clusterings

SRCC, Kendall Rank Correlation Coefficient, KRCC, and Goodman–Kruskal Correlation Coefficient, GKCC. Third, we used efficient consensus functions to combine these independent clusterings into one final consensus clustering. We investigated and compared the effectiveness of three consensus functions: Cluster-Based Graph Formulation, CBGF, Hybrid Bipartite Graph Formulation, HBGF, and Instance-Based Graph Formulation, IBGF. Fourth, in order to enable processing of large data sets and new data, the resulting consensus clustering of the final randomized sample was used as a training set to train fast supervised classification algorithms. These fast classification algorithms were then used to classify the whole large data set.

Our experiments compared the effectiveness of CBGF, HBGF and IBGF consensus functions in conjunction with various classification algorithms. The experimental results have shown that the combination of the Goodman–Kruskal Correlation Coefficient, Hybrid Bipartite Graph Formulation consensus function and the Sequential Minimal Optimization classifier with the polynomial kernel achieved the best precision and recall in this scheme. This combination can be recommended for future implementations and applications for profiling of very large data sets of phishing websites in order to prepare data for subsequent forensic analysis based on the resulting individual clusters.

ACKNOWLEDGMENT

The second author was supported by Queen Elizabeth II Fellowship, Discovery grant DP0211866 and Linkage grant LP0990908 from Australian Research Council. The third author was supported by ARC Discovery grant DP0449469. All authors were supported by a Ballarat-Deakin collaboration grant.

The authors are grateful to three referees for thorough reports with detailed comments and corrections, which have helped to improve the text of this article.

REFERENCES

[1] R. Dazeley, J. Yearwood, B. Kang, and A. Kelarev, "Consensus clustering and supervised classification for profiling

TABLE I.
PRECISION OF SMO, LIBSVM AND LIBLINEAR FOR PLCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	82.523	86.558	83.262
polynomial kernel	83.117	89.368	85.394
Pearson universal	79.795	87.196	85.061
RBFKernel	72.409	80.264	75.785
LibSVM C-SVC			
linear kernel	62.305	71.306	67.385
polynomial kernel	63.973	69.451	69.735
radial basis function	63.944	65.731	66.780
sigmoid kernel	5.358	1.430	3.272
LibSVM nu-SVC			
linear kernel	61.829	72.097	69.555
polynomial kernel	60.229	59.812	60.934
radial basis function	51.084	54.907	54.476
sigmoid kernel	0.283	2.102	0.066
LibLINEAR			
L2 loss svm (dual)	36.424	39.393	37.098
L1 loss svm (dual)	36.946	45.355	40.509
multi-class svm	55.893	62.003	57.362

TABLE II.
RECALL OF SMO, LIBSVM AND LIBLINEAR FOR PLCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	82.525	86.558	83.260
polynomial kernel	83.118	89.370	85.396
Pearson universal	79.795	87.194	85.061
RBFKernel	72.408	80.265	75.786
LibSVM C-SVC			
linear kernel	62.307	71.304	67.383
polynomial kernel	63.974	69.453	69.737
radial basis function	63.941	65.729	66.781
sigmoid kernel	5.359	1.432	3.273
LibSVM nu-SVC			
linear kernel	61.830	72.096	69.554
polynomial kernel	60.229	59.812	60.936
radial basis function	51.082	54.908	54.475
sigmoid kernel	0.284	2.100	0.065
LibLINEAR			
L2 loss svm (dual)	36.424	39.393	37.098
L1 loss svm (dual)	36.948	45.356	40.507
multi-class svm	55.893	62.001	57.362

TABLE III.
PRECISION OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR PLCC

	CBGF	HBGF	IBGF
BayesNet	70.541	76.874	75.147
DecisionTable	65.923	71.955	64.624
IBk	78.091	83.981	80.987
J48	70.545	80.670	76.027
JRip	70.733	78.964	71.389
HyperPipes	54.593	56.560	56.732
LibLINEAR	55.893	62.003	57.362
LibSVM	63.973	69.451	69.735
NaiveBayes	62.546	66.328	64.041
PART	67.932	78.082	75.298
RBFNetwork	53.755	58.741	55.985
Ridor	62.905	68.268	61.313
SMO	83.117	89.368	85.394
VFI	58.227	62.603	59.409

TABLE IV.
RECALL OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR PLCC

	CBGF	HBGF	IBGF
BayesNet	70.541	76.875	75.147
DecisionTable	65.925	71.956	64.623
IBk	78.089	83.982	80.989
J48	70.542	80.671	76.026
JRip	70.733	78.964	71.388
HyperPipes	54.592	56.558	56.732
LibLINEAR	55.893	62.001	57.362
LibSVM	63.974	69.453	69.737
NaiveBayes	62.544	66.326	64.042
PART	67.932	78.081	75.296
RBFNetwork	53.756	58.741	55.985
Ridor	62.905	68.269	61.311
SMO	83.118	89.370	85.396
VFI	58.229	62.603	59.407

TABLE V.
PRECISION OF SMO, LIBSVM AND LIBLINEAR FOR SRCC

	CBGF	HBGF	IBGF
SMO			
- normalized polynomial	84.770	91.331	88.101
- polynomial kernel	87.441	94.759	90.726
- Pearson universal	84.236	91.936	87.557
- RBFKernel	76.766	82.844	79.028
LibSVM C-SVC			
- linear kernel	68.173	74.573	71.797
- polynomial kernel	69.648	74.671	72.252
- radial basis function	65.949	71.525	68.961
- sigmoid kernel	2.506	2.042	1.883
LibSVM nu-SVC			
- linear kernel	67.515	72.831	70.459
- polynomial kernel	60.460	65.254	63.405
- radial basis function	55.358	60.206	57.913
- sigmoid kernel	2.290	2.113	2.591
LibLINEAR			
- L2 loss svm (dual)	40.040	43.750	41.823
- L1 loss svm (dual)	41.113	44.675	42.891
- multi-class svm	61.012	65.286	63.267

phishing emails in internet commerce security," in *Knowledge Management and Acquisition for Smart Systems and Services, PKAW2010, Lecture Notes in Computer Science*, vol. 6232, 2010, pp. 235–246.

[2] J. Yearwood, D. Webb, L. Ma, P. Vamplew, B. Ofoghi, and A. Kelarev, "Applying clustering and ensemble clustering approaches to phishing profiling," in *Data Mining and Analytics 2009, Proc. 8th Australasian Data Mining Conference: AusDM 2009, CRPIT*, vol. 101, 2009, pp. 25–34.

[3] APWG, "Anti-Phishing Working Group," urlhttp://apwg.org/, accessed 15 December 2010.

[4] OECD, "Organisation for Economic Cooperation and Development, OECD task force on spam, OECD anti-spam toolkit and its annexes," http://www.oecd.org/dataoecd/63/28/36494147.pdf, accessed 20 November 2011.

[5] PhishTank, "Developer information," http://www.phishtank.com/developer_info.php, viewed 20 September 2011.

[6] T. Joachims, "A probabilistic analysis of the rocchio algorithm with TF-IDF for text categorization," in *Proc. 14th International Conference on Machine Learning*, 1997, pp. 143–151.

[7] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Dordrecht: Kluwer, 1988.

[8] NIST/SEMATECH, "E-handbook of statistical methods," http://www.itl.nist.gov/div898/handbook/, viewed 21 October 2011.

[9] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[10] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Amsterdam: Elsevier/Morgan Kaufman, 2005.

[11] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[12] D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 1987.

[13] J. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artificial Intelligence*, vol. 40, pp. 11–61, 1990.

[14] S. Hochbaum, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, pp. 180–184, 1985.

[15] P. Rousseeuw, "Silhouettes: a graphical aid to the interpre-

TABLE VI.
RECALL OF SMO, LIBSVM AND LIBLINEAR FOR SRCC

	CBGF	HBGF	IBGF
SMO			
- normalized polynomial	84.763	91.326	88.096
- polynomial kernel	87.443	94.760	90.727
- Pearson universal	84.222	91.921	87.542
- RBFKernel	76.707	82.783	78.966
LibSVM C-SVC			
- linear kernel	68.185	74.585	71.806
- polynomial kernel	69.608	74.632	72.216
- radial basis function	65.697	71.270	68.709
- sigmoid kernel	2.624	2.159	2.000
LibSVM nu-SVC			
- linear kernel	67.494	72.809	70.439
- polynomial kernel	60.534	65.329	63.477
- radial basis function	55.219	60.064	57.772
- sigmoid kernel	2.404	2.229	2.709
LibLINEAR			
- L2 loss svm (dual)	39.995	43.703	41.779
- L1 loss svm (dual)	41.041	44.606	42.819
- multi-class svm	61.004	65.274	63.259

TABLE VII.
PRECISION OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR SRCC

	CBGF	HBGF	IBGF
BayesNet	73.952	81.112	77.297
DecisionTable	67.505	73.394	70.635
IBk	80.792	87.195	83.775
J48	76.266	83.397	79.640
JRip	74.195	81.131	77.477
HyperPipes	55.380	59.686	57.198
LibLINEAR	61.012	65.286	63.267
LibSVM	69.648	74.671	72.252
NaiveBayes	66.115	71.053	68.312
PART	73.584	79.476	76.201
RBFNetwork	58.726	63.716	61.639
Ridor	64.307	69.406	66.633
SMO	87.441	94.759	90.726
VFI	61.678	66.935	64.590

TABLE VIII.
RECALL OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR SRCC

	CBGF	HBGF	IBGF
BayesNet	73.941	81.103	77.287
DecisionTable	67.505	73.392	70.633
IBk	80.795	87.196	83.774
J48	76.269	83.397	79.644
JRip	74.195	81.131	77.476
HyperPipes	55.298	59.605	57.121
LibLINEAR	61.004	65.274	63.259
LibSVM	69.608	74.632	72.216
NaiveBayes	66.110	71.046	68.307
PART	73.582	79.473	76.195
RBFNetwork	58.726	63.715	61.641
Ridor	64.312	69.406	66.638
SMO	87.443	94.760	90.727
VFI	61.660	66.918	64.570

TABLE IX.
PRECISION OF SMO, LIBSVM AND LIBLINEAR FOR KRCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	82.275	89.446	86.392
polynomial kernel	85.035	93.313	90.199
Pearson universal	82.154	90.692	86.994
RBFKernel	74.100	82.185	80.238
LibSVM C-SVC			
linear kernel	69.852	73.033	68.161
polynomial kernel	69.914	74.748	69.517
radial basis function	65.649	72.981	67.877
sigmoid kernel	5.349	3.350	2.336
LibSVM nu-SVC			
linear kernel	68.858	71.027	68.027
polynomial kernel	59.830	63.442	62.252
radial basis function	56.572	60.520	54.895
sigmoid kernel	4.735	2.159	4.154
LibLINEAR			
L2 loss svm (dual)	41.196	44.240	39.293
L1 loss svm (dual)	39.862	43.767	44.358
multi-class svm	61.479	62.475	65.171

tation and validation of cluster analysis," *J. Comp. Appl. Math.*, vol. 20, pp. 53–65, 1987.

[16] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley, 2005.

[17] X. Fern and C. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *21st International Conference on Machine Learning, ICML'04*, vol. 69. New York, NY, USA: ACM, 2004, pp. 36–43.

[18] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *J. Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[19] A. Topchy, A. Jain, and W. Punch, "Combining multiple weak clusterings," in *IEEE International Conference on*

TABLE X.
RECALL OF SMO, LIBSVM AND LIBLINEAR FOR KRCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	82.274	89.447	86.390
polynomial kernel	85.035	93.314	90.201
Pearson universal	82.153	90.693	86.995
RBFKernel	74.101	82.184	80.237
LibSVM C-SVC			
linear kernel	69.850	73.032	68.160
polynomial kernel	69.915	74.746	69.517
radial basis function	65.649	72.982	67.878
sigmoid kernel	5.348	3.350	2.337
LibSVM nu-SVC			
linear kernel	68.861	71.028	68.026
polynomial kernel	59.831	63.442	62.252
radial basis function	56.573	60.519	54.897
sigmoid kernel	4.733	2.158	4.154
LibLINEAR			
L2 loss svm (dual)	41.195	44.241	39.294
L1 loss svm (dual)	39.861	43.766	44.359
multi-class svm	61.480	62.474	65.170

TABLE XI.
PRECISION OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR KRCC

	CBGF	HBGF	IBGF
BayesNet	71.738	78.649	74.681
DecisionTable	69.551	69.676	70.251
IBk	81.610	87.794	82.920
J48	75.302	81.006	78.669
JRip	74.918	80.225	77.095
HyperPipes	56.481	58.957	55.298
LibLINEAR	61.479	62.475	65.171
LibSVM	69.914	74.748	69.517
NaiveBayes	63.786	68.185	70.127
PART	74.310	75.865	75.760
RBFNetwork	57.941	64.090	63.067
Ridor	61.343	69.917	68.312
SMO	85.035	93.313	90.199
VFI	60.715	64.436	60.965

TABLE XII.
RECALL OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR KRCC

	CBGF	HBGF	IBGF
BayesNet	71.738	78.649	74.682
DecisionTable	69.552	69.678	70.252
IBk	81.609	87.795	82.919
J48	75.304	81.007	78.671
JRip	74.916	80.225	77.095
HyperPipes	56.481	58.957	55.300
LibLINEAR	61.480	62.474	65.170
LibSVM	69.915	74.746	69.517
NaiveBayes	63.787	68.185	70.128
PART	74.310	75.863	75.760
RBFNetwork	57.939	64.092	63.069
Ridor	61.341	69.916	68.315
SMO	85.035	93.314	90.201
VFI	60.716	64.435	60.963

TABLE XIII.
PRECISION OF SMO, LIBSVM AND LIBLINEAR FOR GKCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	87.941	94.039	90.871
polynomial kernel	89.460	96.798	93.222
Pearson universal	85.122	95.096	88.679
RBFKernel	78.120	84.699	79.237
LibSVM C-SVC			
linear kernel	68.890	79.445	71.253
polynomial kernel	74.226	77.030	73.931
radial basis function	69.038	72.425	69.297
sigmoid kernel	3.901	3.015	0.039
LibSVM nu-SVC			
linear kernel	72.314	76.149	73.842
polynomial kernel	59.326	66.251	63.978
radial basis function	58.169	61.328	60.879
sigmoid kernel	3.109	2.215	4.108
LibLINEAR			
L2 loss svm (dual)	39.844	42.143	43.905
L1 loss svm (dual)	39.597	47.726	41.420
multi-class svm	65.681	69.315	64.760

Data Mining, 2003, pp. 331–338.

[20] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Centre, Minneapolis, Technical Report, 1998.

[21] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[22] D. Sridhar, E. Bartlett, and R. Seagrave, "Information theoretic subset selection for neural network models," *Computers & Chemical Engineering*, vol. 22, pp. 613–626, 1998.

TABLE XIV.
RECALL OF SMO, LIBSVM AND LIBLINEAR FOR GKCC

	CBGF	HBGF	IBGF
SMO			
normalized polynomial	86.689	93.310	90.712
polynomial kernel	89.306	97.761	94.127
Pearson universal	86.759	93.942	89.253
RBFKernel	77.288	83.634	82.719
LibSVM C-SVC			
linear kernel	68.550	74.427	73.851
polynomial kernel	69.490	77.033	77.401
radial basis function	69.171	70.977	69.556
sigmoid kernel	0.341	1.541	4.773
LibSVM nu-SVC			
linear kernel	70.052	75.046	74.891
polynomial kernel	60.537	69.804	65.771
radial basis function	58.797	64.382	58.831
sigmoid kernel	3.284	3.417	2.170
LibLINEAR			
L2 loss svm (dual)	39.897	44.316	45.019
L1 loss svm (dual)	39.350	46.340	44.643
multi-class svm	63.726	64.615	63.241

TABLE XV.
PRECISION OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR GKCC

	CBGF	HBGF	IBGF
BayesNet	76.754	84.991	82.397
DecisionTable	70.413	78.514	72.500
IBk	81.541	87.845	86.408
J48	77.093	85.858	80.078
JRip	76.432	81.829	79.472
HyperPipes	59.988	62.316	60.631
LibLINEAR	65.681	69.315	64.760
LibSVM	74.226	77.030	73.931
NaiveBayes	67.036	75.893	71.430
PART	75.048	79.226	77.220
RBFNetwork	60.633	64.450	64.452
Ridor	63.499	73.403	65.850
SMO	89.460	96.798	93.222
VFI	64.419	67.442	68.917

TABLE XVI.
RECALL OF CLASSIFIERS WITH CBGF, HBGF, IBGF FOR GKCC

	CBGF	HBGF	IBGF
BayesNet	76.752	84.989	82.397
DecisionTable	70.414	78.516	72.502
IBk	81.542	87.847	86.408
J48	77.093	85.858	80.077
JRip	76.430	81.826	79.474
HyperPipes	59.987	62.317	60.629
LibLINEAR	63.726	64.615	63.241
LibSVM	69.490	77.033	77.401
NaiveBayes	67.035	75.893	71.431
PART	75.049	79.226	77.219
RBFNetwork	60.630	64.448	64.452
Ridor	63.499	73.401	65.850
SMO	89.306	97.761	94.127
VFI	64.417	67.441	68.915

[23] Y. Hong, S. Kwong, Y. Chang, and Q. Ren, "Consensus unsupervised feature ranking from multiple views," *Pattern Recognition Letters*, vol. 29, pp. 595–602, 2008.

[24] G. Corder and D. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New York: Wiley Interscience, 2009.

[25] M. Kendall and J. Gibbons, *Rank Correlation Methods*, 5th ed. London: Oxford University Press, 1990.

[26] R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse, "Weka manual for version 3-7-3," urlhttp://www.cs.waikato.ac.nz/ml/weka/, viewed 15 August 2011.

[27] R. Kohavi, "The power of decision tables," in *8th European Conference on Machine Learning*, 1995, pp. 174–189.

[28] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[29] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[30] W. Cohen, "Fast effective rule induction," in *Proc. 12th Internat. Conf. Machine Learning*, 1995, pp. 115–123.

[31] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear - a library for large linear classification," Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/, viewed 10 August 2011.

[32] C.-C. Chang and C.-J. Lin, "Libsvm – a library for support vector machines," Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/, viewed 12 June 2011, 2001.

[33] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training svm," *J. Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[34] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[35] F. Frank and I. Witten, "Generating accurate rule sets without global optimization," in *Proc. 15th Internat. Conf. on Machine Learning*, 1998, pp. 144–151.

[36] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems*, 1998.

[37] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[38] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, 1998.

[39] G. Demiroz and A. Guvenir, "Classification by voting feature intervals," in *Proc. 9th European Conference on Machine Learning*, 1997, pp. 85–92.

[40] X. Wu, V. Kumar, J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge Inf. Systems*, vol. 14, pp. 1–37, 2007.

[41] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Dept. Computer Science, National Taiwan University, http://www.csie.ntu.edu.tw/~cjlin, Initial version: 2003, last updated: April 15, 2010.

[42] G. Beliakov and J. Ugon, "Implementation of novel methods of global and non-smooth optimization: GANSO programming library," *Optimization*, vol. 56, pp. 543–546, 2007.